

52.1 OR—Logical Inclusive OR

Opcode	Instruction	Description
0C <i>ib</i>	OR AL, <i>imm8</i>	AL OR <i>imm8</i>
0D <i>iw</i>	OR AX, <i>imm16</i>	AX OR <i>imm16</i>
0D <i>id</i>	OR EAX, <i>imm32</i>	EAX OR <i>imm32</i>
80 /1 <i>ib</i>	OR <i>r/m8</i> , <i>imm8</i>	<i>r/m8</i> OR <i>imm8</i>
81 /1 <i>iw</i>	OR <i>r/m16</i> , <i>imm16</i>	<i>r/m16</i> OR <i>imm16</i>
81 /1 <i>id</i>	OR <i>r/m32</i> , <i>imm32</i>	<i>r/m32</i> OR <i>imm32</i>
83 /1 <i>ib</i>	OR <i>r/m16</i> , <i>imm8</i>	<i>r/m16</i> OR <i>imm8</i> (sign-extended)
83 /1 <i>ib</i>	OR <i>r/m32</i> , <i>imm8</i>	<i>r/m32</i> OR <i>imm8</i> (sign-extended)
08 /r	OR <i>r/m8</i> , <i>r8</i>	<i>r/m8</i> OR <i>r8</i>
09 /r	OR <i>r/m16</i> , <i>r16</i>	<i>r/m16</i> OR <i>r16</i>
09 /r	OR <i>r/m32</i> , <i>r32</i>	<i>r/m32</i> OR <i>r32</i>
0A /r	OR <i>r8</i> , <i>r/m8</i>	<i>r8</i> OR <i>r/m8</i>
0B /r	OR <i>r16</i> , <i>r/m16</i>	<i>r16</i> OR <i>r/m16</i>
0B /r	OR <i>r32</i> , <i>r/m32</i>	<i>r32</i> OR <i>r/m32</i>

Description

Performs a bitwise inclusive OR operation between the destination (first) and source (second) operands and stores the result in the destination operand location. The source operand can be an immediate, a register, or a memory location; the destination operand can be a register or a memory location. (However, two memory operands cannot be used in one instruction.) Each bit of the result of the OR instruction is 0 if both corresponding bits of the operands are 0; otherwise, each bit is 1.

Operation

DEST ← *DEST* OR *SRC*;

Flags Affected

The OF and CF flags are cleared; the SF, ZF, and PF flags are set according to the result. The state of the AF flag is undefined.

Protected Mode Exceptions

- | | |
|-----------------|---|
| #GP(0) | If the destination operand points to a nonwritable segment. |
| | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| | If the DS, ES, FS, or GS register contains a null segment selector. |
| #SS(0) | If a memory operand effective address is outside the SS segment limit. |
| #PF(fault-code) | If a page fault occurs. |

#AC(0) If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3.

Real-Address Mode Exceptions

#GP If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.

#SS If a memory operand effective address is outside the SS segment limit.

Virtual-8086 Mode Exceptions

#GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit.

#SS(0) If a memory operand effective address is outside the SS segment limit.

#PF(fault-code) If a page fault occurs.

#AC(0) If alignment checking is enabled and an unaligned memory reference is made.

52.2 OUT—Output to Port

Opcode	Instruction	Description
E6 <i>ib</i>	OUT <i>imm8</i> , AL	Output byte in AL to I/O port address <i>imm8</i>
E7 <i>ib</i>	OUT <i>imm8</i> , AX	Output word in AX to I/O port address <i>imm8</i>
E7 <i>ib</i>	OUT <i>imm8</i> , EAX	Output doubleword in EAX to I/O port address <i>imm8</i>
EE	OUT DX, AL	Output byte in AL to I/O port address in DX
EF	OUT DX, AX	Output word in AX to I/O port address in DX
EF	OUT DX, EAX	Output doubleword in EAX to I/O port address in DX

Description

Copies the value from the second operand (source operand) to the I/O port specified with the destination operand (first operand). The source operand can be register AL, AX, or EAX, depending on the size of the port being accessed (8, 16, or 32 bits, respectively); the destination operand can be a byte-immediate or the DX register. Using a byte immediate allows I/O port addresses 0 to 255 to be accessed; using the DX register as a source operand allows I/O ports from 0 to 65,535 to be accessed.

The size of the I/O port being accessed is determined by the opcode for an 8-bit I/O port or by the operand-size attribute of the instruction for a 16- or 32-bit I/O port.

At the machine code level, I/O instructions are shorter when accessing 8-bit I/O ports. Here, the upper eight bits of the port address will be 0.

This instruction is only useful for accessing I/O ports located in the processor's I/O address space. See “Input/Output”, for more information on accessing I/O ports in the I/O address space.

Intel Architecture Compatibility

After executing an OUT instruction, the Pentium processor insures that the EWBE# pin has been sampled active before it begins to execute the next instruction. (Note that the instruction can be prefetched if EWBE# is not active, but it will not be executed until the EWBE# pin is sampled active.) Only the Pentium processor family has the EWBE# pin; the other Intel Architecture processors do not.

Operation

```
IF ((PE = 1) AND ((CPL > IOPL) OR (VM = 1)))
    THEN (* Protected mode with CPL > IOPL or virtual-8086 mode *)
        IF (Any I/O Permission Bit for I/O port being accessed = 1)
            THEN (* I/O operation is not allowed *)
                #GP(0);
            ELSE (* I/O operation is allowed *)
                DEST ← SRC; (* Writes to selected I/O port *)
        FI;
    ELSE (Real Mode or Protected Mode with CPL ≤ IOPL *)
        DEST ← SRC; (* Writes to selected I/O port *)
    FI;
```

Flags Affected

None.

Protected Mode Exceptions

#GP(0) If the CPL is greater than (has less privilege) the I/O privilege level (IOPL) and any of the corresponding I/O permission bits in TSS for the I/O port being accessed is 1.

Real-Address Mode Exceptions

None.

Virtual-8086 Mode Exceptions

#GP(0) If any of the I/O permission bits in the TSS for the I/O port being accessed is 1.

52.3 OUTS/OUTSB/OUTSW/OUTSD—Output String to Port

Opcode	Instruction	Description
6E	OUTS DX, m8	Output byte from memory location specified in DS:(E)SI to I/O port specified in DX
6F	OUTS DX, m16	Output word from memory location specified in DS:(E)SI to I/O port specified in DX
6F	OUTS DX, m32	Output doubleword from memory location specified in DS:(E)SI to I/O port specified in DX
6E	OUTSB	Output byte from memory location specified in DS:(E)SI to I/O port specified in DX
6F	OUTSW	Output word from memory location specified in DS:(E)SI to I/O port specified in DX
6F	OUTSD	Output doubleword from memory location specified in DS:(E)SI to I/O port specified in DX

Description

Copies data from the source operand (second operand) to the I/O port specified with the destination operand (first operand). The source operand is a memory location, the address of which is read from either the DS:EDI or the DS:DI registers (depending on the address-size attribute of the instruction, 32 or 16, respectively). (The DS segment may be overridden with a segment override prefix.) The destination operand is an I/O port address (from 0 to 65,535) that is read from the DX register. The size of the I/O port being accessed (that is, the size of the source and destination operands) is determined by the opcode for an 8-bit I/O port or by the operand-size attribute of the instruction for a 16- or 32-bit I/O port.

At the assembly-code level, two forms of this instruction are allowed: the “explicit-operands” form and the “no-operands” form. The explicit-operands form (specified with the OUTS mnemonic) allows the source and destination operands to be specified explicitly. Here, the source operand should be a symbol that indicates the size of the I/O port and the source address, and the destination operand must be DX. This explicit-operands form is provided to allow documentation; however, note that the documentation provided by this form can be misleading. That is, the source operand symbol must specify the correct **type** (size) of the operand (byte, word, or doubleword), but it does not have to specify the correct **location**. The location is always specified by the DS:(E)SI registers, which must be loaded correctly before the OUTS instruction is executed.

The no-operands form provides “short forms” of the byte, word, and doubleword versions of the OUTS instructions. Here also DS:(E)SI is assumed to be the source operand and DX is assumed to be the destination operand. The size of the I/O port is specified with the choice of mnemonic: OUTSB (byte), OUTSW (word), or OUTSD (doubleword).

After the byte, word, or doubleword is transferred from the memory location to the I/O port, the (E)SI register is incremented or decremented automatically according to the setting of the DF flag in the EFLAGS register. (If the DF flag is 0, the (E)SI register is incremented; if the DF flag is 1, the (E)SI register is decremented.) The (E)SI register is incremented or decremented by 1 for byte operations, by 2 for word operations, or by 4 for doubleword operations.

The OUTS, OUTSB, OUTSW, and OUTSD instructions can be preceded by the REP prefix for block input of ECX bytes, words, or doublewords. See “REP/REPE/REPZ/REPNE /REPNZ—Repeat String Operation Prefix” in this chapter for a description of the REP prefix.

This instruction is only useful for accessing I/O ports located in the processor’s I/O address space. See “Input/Output”, for more information on accessing I/O ports in the I/O address space.

Intel Architecture Compatibility

After executing an OUTS, OUTSB, OUTSW, or OUTSD instruction, the Pentium processor insures that the EWBE# pin has been sampled active before it begins to execute the next instruction. (Note that the instruction can be prefetched if EWBE# is not active, but it will not be executed until the EWBE# pin is sampled active.) Only the Pentium processor family has the EWBE# pin; the other Intel Architecture processors do not.

Operation

```
IF ((PE = 1) AND ((CPL > IOPL) OR (VM = 1)))
  THEN (* Protected mode with CPL > IOPL or virtual-8086 mode *)
    IF (Any I/O Permission Bit for I/O port being accessed = 1)
      THEN (* I/O operation is not allowed *)
        #GP(0);
      ELSE (* I/O operation is allowed *)
        DEST ← SRC; (* Writes to I/O port *)
    FI;
ELSE (Real Mode or Protected Mode with CPL ≤ IOPL *)
  DEST ← SRC; (* Writes to I/O port *)
```

```

FI;
IF (byte transfer)
    THEN IF DF = 0
        THEN (E)SI ← (E)SI + 1;
        ELSE (E)SI ← (E)SI - 1;
    FI;
ELSE IF (word transfer)
    THEN IF DF = 0
        THEN (E)SI ← (E)SI + 2;
        ELSE (E)SI ← (E)SI - 2;
    FI;
ELSE (* doubleword transfer *)
    THEN IF DF = 0
        THEN (E)SI ← (E)SI + 4;
        ELSE (E)SI ← (E)SI - 4;
    FI;
FI;
FI;

```

Flags Affected

None.

Protected Mode Exceptions

- | | |
|-----------------|--|
| #GP(0) | <p>If the CPL is greater than (has less privilege) the I/O privilege level (IOPL) and any of the corresponding I/O permission bits in TSS for the I/O port being accessed is 1.</p> <p>If a memory operand effective address is outside the limit of the CS, DS, ES, FS, or GS segment.</p> <p>If the segment register contains a null segment selector.</p> |
| #PF(fault-code) | If a page fault occurs. |
| #AC(0) | If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3. |

Real-Address Mode Exceptions

- | | |
|-----|---|
| #GP | If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. |
| #SS | If a memory operand effective address is outside the SS segment limit. |

Virtual-8086 Mode Exceptions

- | | |
|-----------------|--|
| #GP(0) | If any of the I/O permission bits in the TSS for the I/O port being accessed is 1. |
| #PF(fault-code) | If a page fault occurs. |
| #AC(0) | If alignment checking is enabled and an unaligned memory reference is made. |

